A Component Framework Supporting Peer Services for Space Data Management

Daniel Crichton
Jet Propulsion Laboratory
4800 Oak Grove Dr 171-264
Pasadena, CA 91109
Dan.Crichton@jpl.nasa.gov

Steve Hughes
Jet Propulsion Laboratory
4800 Oak Grove Dr 171-264
Pasadena, CA 91109
Steven.Hughes@jpl.nasa.gov

Sean Kelly
Independent Consultant
1672 Hope Dr #1817
Santa Clara, CA 95054
Sean.Kelly@jpl.nasa.gov

Paul Ramirez
Jet Propulsion Laboratory
4800 Oak Grove Dr 171-264
Pasadena, CA 91109
Paul.Ramirez@jpl.nasa.gov

Abstract—Knowledge discovery and data correlation require a unified approach to basic data management. However, achieving such an approach is nearly impossible with hundreds of disparate data sources, legacy systems and data formats. This problem is pervasive in the space science community where data models, taxonomies and data management systems are locally implemented and limited metadata has been collected and organized. Technology developed by the Object Oriented Data Technology (OODT) task at NASA's Jet Propulsion Laboratory (JPL) has been exploring component frameworks for managing, locating and exchanging data residing within a geographically distributed network. OODT has taken a novel approach towards solving this problem by exploiting web technologies usually dedicated to e-commerce, combined with a rich, metadata-based environment. The components developed by OODT create a set of distributed peer-to-peer services that allow for data managed by a peer to be searched and returned as part of an integrated data management system. This paper discusses the approach taken to develop a software framework, and two prototype development efforts for the Planetary Data System (PDS) and the Mission and Ground Asset Database.[1]

TABLE OF CONTENTS

1. INTRODUCTION

The Object Oriented Data Technology (OODT) task, supported by the National Aeronautics and Space Administration (NASA) and implemented at the Jet Propulsion Laboratory (JPL), is focusing on developing a reference framework to support cross disciplinary solutions for data access, exchange and distribution of distributed data to increase the collaboration and correlation of key engineering and science data produced throughout the lifecycle of a space mission. NASA is ambitiously planning to fly several new missions with an increasing amount of data planned for each mission. Unfortunately, the data collected from cradle to grave of these missions tends to be difficult to locate, access, understand and use limiting interoperability, software and data reuse, knowledge discovery and increasing mission costs. As a result, OODT identified several key goals including (1) understanding where the data resources reside, (2) understanding how data is accessed in each system, (3) understanding the underlying data models for each system, and (4) building ubiquitous interfaces across multiple data systems that demonstrate interoperability using a common query mechanism. In order to meet these challenging goals we defined an architecture and a set of requirements. The architecture defined a framework, services, and a reference information model that was tested against several projects including the Planetary Data System (PDS) and the Deep Space Network (DSN). We based requirements on experience from these activities as well as interactions with several other efforts which support the overall objectives and culture of JPL and NASA. The partnerships established by the OODT team to research solutions proved to be critical to meeting our goals, and allowed us to identify key data architectures that support the management and exchange of data in an era of ever-increasing disparate data sources.

2. HISTORY AND PROGRAMMATIC GOALS

In February 1998, JPL was funded by the Office of Space Science at NASA to perform research in object oriented and distributed data management technologies. A principal goal of the OODT task is to explore and develop cutting-edge systems for the interoperability of scientific research, measurements, and data that span multiple disciplines. Given the amount of data generated by past unmanned deep space missions, and the predicted quantity of data that is planned for future missions, key information technology solutions need to be created that will enable the archiving and distribution of science data products residing in multiple data systems. As a result, the focus of OODT has been to provide an architectural framework that includes software services for data archiving, data distribution, data location and data exchange. The ability to correlate data across multiple missions, databases and data sets is dependent on having the data management infrastructure to allow the

exchange of data. Advances in technologies including the Extensible Markup Language (XML) [1], metadata management, and distributed computing have allowed such an infrastructure to be a reality.

Presently, space scientists cannot easily locate or use data across the hundreds of autonomous, heterogeneous and distributed data systems within the space science community. The various platforms, data formats, data interpretations and Database Management Systems (DBMS) used make interoperability a difficult task. Developing a ubiquitous computing environment across data systems is a difficult challenge given the independence of work groups and research activities at NASA. Therefore, the architecture that we have defined allows each data system to be independently developed, however, the systems exchange data given a messaging infrastructure that includes metadata[2] descriptions associated with the messages. The architecture is based on a directed graph (or digraph) of data systems that are traversed using a common query message implemented in XML in order to satisfy a query. Profiles—sets of resource definitions—describe nodes (or data systems) that are part of the digraph. Profiles may point to other profiles thus representing arcs of the digraph. A profile is essentially a metadata description of the resources known at a node in the distributed framework. These resources can be definitions of data systems, or data products managed by a data system. Both the service and information architecture will be defined in more detail later in the paper.

As systems become more distributed, the need to integrate dissimilar systems increases. A simple calculation yields that an organization with *N* number of systems could potentially identify the need to create $(N^2-N)/2$ interfaces in order to share data between all of those systems. As a result, the need to identify and support standards based interfaces increases as the number of distributed nodes in an organization increases. One of the emerging solutions is peer-to-peer (P2P) architectures. A P2P architecture is defined as "environment consist[ing] of computers with equal capabilities that share resources (such as processing power and memory), communicate exclusively with each other and do not connect to servers or central databases" [2]. One of the key goals of OODT has been to provide the ability for data systems to become peers that provide services. These services could distribute information such as science data, engineering data, navigation data or even contain computational capabilities such as science processing algorithms or simulation environments. The Internet has spawned several such capabilities including Napster for sharing music files. It is important to note that while it is generally accepted that federating together data systems will increase the knowledge generated by correlating data captured in dispersed space science databases, it is still unclear as to what the return on investment will yield. Several initiatives at NASA are exploring that in addition to OODT including the National Virtual Observatory (NVO) which is attempting to build a federated environment for collaboration and research that integrates astrophysics and astronomy databases across the science community with a goal to "enable new science, more effective science, and more cost effective science" [3]. As a result, one of the activities outlined by NVO is to "[establish] a common systems approach to data pipelining, archiving, and retrieval that will ensure easy access by a large and diverse community of users and that will minimize costs and times to completion." [3] As these efforts continue one goal is clear: the need to have a generic, adaptable, scalable and extensible architecture with well defined interfaces that implement standards based protocols and a rich metadata infrastructure is an important key to unlocking interoperability across NASA.

[2] Metadata is data about data.

3. Peer Services Architecture

OODT consists of a set of cooperating, distributed peer components. Although its current incarnation is implemented with a client-server communications substrate, the design resembles a P2P network, and we have plans to transition the communications substrate to a P2P implementation shortly [4].

Distributed Framework

OODT is a distributed system, wherein components may be geographically dispersed. A standard Internet TCP/IP connection suffices to provide connectivity between components at this time. In addition, we've architected the software as a plug-in system [5]. Rather than make OODT a set of utility APIs out of which each site develops its own applications, we create a set of classes and interfaces that sites customize and implement and then register with the OODT API. Frameworks are easier to use because the framework classes and interfaces tell the developer exactly what needs to be implemented. More work necessarily falls upon the framework developers.

Within OODT there are three major components:

- **Profile servers** serve scientific metadata and can tell whether a particular resource can provide an answer to a query.
- **Product servers** serve data products in a system-independent format when presented with a product request.
- **Query servers** accept profile and product queries and traverse the network of profile and product servers, collecting results.

The query server is the main entry point into the system, with Application Programmer's Interface (API) for

developers and a web interface for end users and testing. In addition to these three servers, an archive server provides a metadata-capable data storage system for any kind of product that can be represented as a sequence of bytes (such as a file).

Common Design Elements

Profile and product servers, in addition to being distributed peer components, exhibit some common design elements. First, they all advertise a remotely accessible interface, callable via CORBA [6]. While clients may contact a profile or product server directly, we encourage them to use the query service which simulates the gateway to the P2P space. Figure 1, a UML [7] deployment diagram, demonstrates a sample deployment of the OODT software. In that diagram, the query server has network connections to two profile and two product servers.

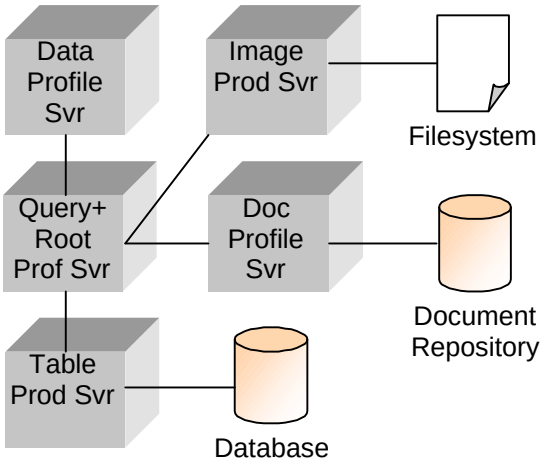

Figure 1 - Sample OODT Deployment

Also, profile and product servers have run-time specified, replaceable backend processors. A profile server may get its metadata from a flat file or a database by changing the backend. Similarly, a product server may retrieve an image from the filesystem or from an image storage device with a customized backend. Backend interfaces mandate the calling conventions for profile and product servers. We achieve this run-time configuration through Java's [8] interface and dynamic loading mechanisms; core product and profile servers refer only to abstract backend interfaces, while specific named classes implement those interfaces.

Finally, profile and product servers all manipulate the same query structure. This structure exists as an XML Document Type Definition and as a Java class, and it encapsulates the user's query plus any results retrieved so far.

Profile Server

The profile server's job is to serve profiles [9], which are XML-representable metadata descriptions of resources. A resource in this context may be a single data granule, a dataset (a collection of granules), a collection of datasets, a data dictionary, anything with a Uniform Resource Identifier (URI) [10], such as a URL, or other profiles in other profile servers. A profile "profiles" a resource by describing data elements it contains, what data dictionary defines those elements, who created it, when, in what language, under what subject headings, and so forth. Section 4 describes the profile information model completely.

Responsibilities—The primary responsibility of a profile server is to accept a query and determine if any of the profiles it manages profile resources that could satisfy that query. Each profile server accepts the query and delegates to a backend implementation to search the set of managed profiles and return any that match. Figure 2 depicts the class architecture in a UML diagram.

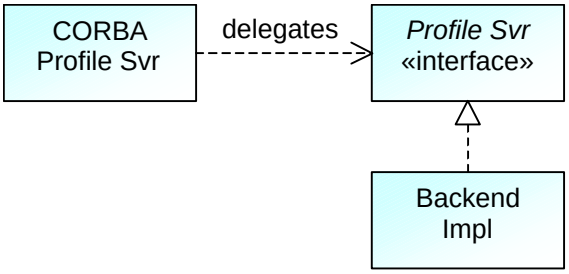

Figure 2 - Profile Service Class Architecture

Although users can contact a specific profile server directly through its CORBA interface, they typically use the query service instead. We bootstrap the query server with a "root" profile server whose set of managed profiles refer to other profile servers. Those other profile servers may in turn refer to yet more profile servers, forming a directed acyclic graph of servers. The query server can crawl this graph (eliminating redundant cyclic queries) and gather results from the entire network. In Figure 1, the query server runs both the query server and the root profile server; the root profile server contains two profiles that refer to the document profile server and the data profile server. The document profile server describes URLs in the document repository, and the data profile server describes the image and table product servers.

Besides searching a set of profiles, each profile server has secondary responsibilities to manage its set of profiles. The management interface includes typical set operations such as adding, modifying, and deleting profiles.

Current Backends—The profile server's plug-in architecture has enabled us to design and experiment with several backend implementations for querying and managing profiles. Because profiles can be represented in XML, our original implementation would store each profile in memory as an XML Document Object Model (DOM) tree, and search by traversing the tree.

Another implementation uses an Oracle relational database, essentially transforming the DOM tree into a series of

database tables. This backend transforms the user's query into SQL and uses a stored procedure to retrieve matching profiles in XML format.

Highly specific applications of profiles are possible, too. At JPL, Xerox's DocuShare [11] product manages large collections of documentation. A profile server backend that uses DocuShare's WebDAV -like interface [12] enables a single query to span multiple DocuSh are and other documentation management systems, enhancing knowledge sharing. In this backend, profiles of resources are built on-the-fly by assembling information from the documentation management system.

Product Server

Product servers exist to provide an interface to retrieving specific scientific data products. Like profile servers, product servers accept the query structure mentioned earlier and described in Section 4. Instead of adding profiles to the results section of t he query structure, they add actual data products. Data products in this sense may be individual granules, datasets, or collections of datasets, depending on the backend implementation to which the product server delegates its query requests.

As with profile servers, product servers delegate requests for products to a specific backend implementation. The backend's job is to act as the translator between a system-dependent product storage system and the system-independent OODT framework. It accepts a quer y, transforms it into a format required by the product storage system, retrieves the product, transforms it into a system-independent format, and adds it to the query structure. Figure 3 shows the class architecture.

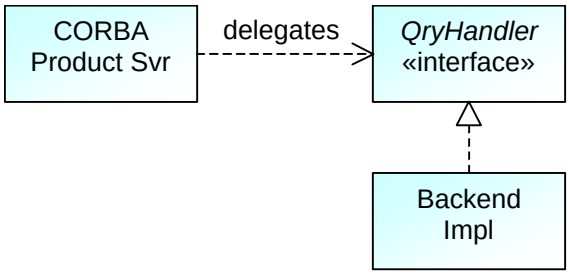

Figure 3 - Product Service Class Architecture

The query handler interface for product servers is far simpler than the profile server interface for profile servers. There is a single method that the backend must imp lement that accepts the generic query structure and returns it with or without matching products installed.

Typical Operation —Users usually won't know ahead of time what product servers exists and by what network names they're known. Likewise, the query s erver does not know what product servers are available either —it's bootstrapped with only a single profile server, the "root" profile server. Some profile servers in the network must contain matching profiles that describe product servers that can answer a request for data.

As a result, querying is a two step process, depicted in the UML interaction diagram in Figure 4. First, the user issues a profile query for the desired data. The query service crawls through the network of p rofile servers and returns any matching profiles. The user then selects a product server from the matching set and reissues the query as a product query directed at a specific server. The query service contacts the product server, which retrieves the product.

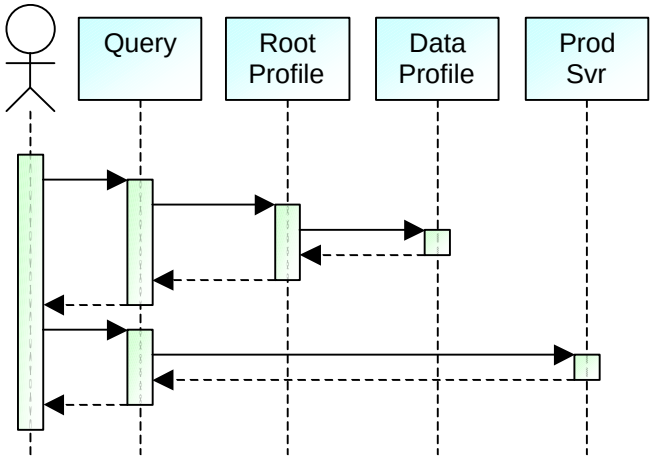

Figure 4 - Retrieving a Product

Product Formats—As mentioned, one of the responsibilities of the backend query handler for a product server is to translate products from system-dependent to a syste m-independent format. These system-independent formats are any that can be encoded with an Internet MIME type [13]. The query structure's results section includes the MIME type of the product. For example, a product server t hat retrieves images from a database in a proprietary format may choose to return them in either the image/jpeg or image/png MIME formats. A tabular product might be in the text/tab-separated-values format.

The OODT framework provides an automated mechani sm to encoded the product itself based on the MIME type. Because the query structure is serialized into XML between peer components, the framework encodes certain MIME types to be compatible with XML's string representation. For example, the framework en codes a text/tab-separated-values product into an XML CDATA section. It encodes binary products like image/gif or application/ms-word into the base-64 textual format [13].

Users have some control over what format product serve rs return products to them. Part of the query structure includes a list of acceptable MIME types. By default, this list is */*, meaning any product format is acceptable. By setting this to a list of preferred MIME types, product servers learn of the users' desires and may encode products appropriately. For example, setting the list to image/jpeg and image/* means

the user prefers images in JPEG format but will accept any image format if necessary. Setting it to image/jpeg alone means only JPEG format i mages are acceptable. When a product server cannot satisfy a user's format request, it may return no product even if it otherwise had a match for the query.

Query Server

A query server provides the point of entry into the OODT framework. The query server contains the algorithms necessary to transform the physical client/server CORBA model into the logical P2P model for executing queries and gathering results. It also provides a simplified interface for users: they don't need to know what profile or produ ct servers exist and what their object addresses are —they only need to access the query server.

Users can access the query server in a variety of ways depending on their sophistication and implementation language. CORBA -capable users can access the query server's CORBA interface directly and call methods to search for profiles or products. Java users may use an API class that encapsulates the CORBA communication. Users of other languages may use two different HTTP [14] interfaces, built using Java Servlets and servlet -capable web servers [15].

One HTTP interface is completely generic: it provides the profile and product search capabilities of the query server through an HTTP GET or POST request. T he user contacts the query URL, passing the XML query structure and other parameters, and receiving the XML query structure back, possibly with encoded results. This interface requires that the client be capable of processing XML documents (and base-64 e ncoded data for binary products) in order to retrieve results from the query structure.

The second HTTP interface provides simplified product searches only and requires no XML processing capability on the client side. The user again uses an HTTP GET or PO ST request, passing a query string and receives back a single product in its system-independent MIME type.

4. INFORMATION ARCHITECTURE

Space science instruments and experiments generate science data products that are archived in science data systems. Effective search and retrieval of data products within and across these data systems depend crucially on information architectures that are organized on principles known to the prospective users and designed to reflect relevant relationships. Well -designed inform ation architectures also enable data browsing, data mining, and correlative science.

Unfortunately, most science data systems were developed using differing organizational principles resulting in heterogeneous information architectures. This makes the search and retrieval of data products across these systems difficult since users are required to connect to each individual data system and deal with dissimilar and often unfamiliar organizing principles.

Metadata organization and management is a key information architectural concept. Metadata engineering also provides key organizational principles and includes data modeling, data element definition, and data resource description. Since metadata has also been identified as key to enabling interoperability betw een heterogeneous data system, the OODT information architecture includes a reference model for describing metadata.

A key feature of the OODT information architecture is the use of commonly accepted standards. We use the ISO/IEC 11179 [18] specification for the definition of data elements in the data element registry. ISO/IEC 11179 is a framework for the specification and standardization of data elements and is widely accepted, providing the underlying basis for data element definition and classification. The specification defines four data element attribute categories: identification, definitional, representational, and administrative. In addition, we've adopted use of the Dublin Core [17] element set, a content description model that has received wide-spread acceptance across the electronic information community. These elements were designed to facilitate the discovery of electronic resources across the Internet and include content, property, and instantiation resource attributes.

We've also adopted the concept of the Object Identifier (OID) to provide persistent, unique, and simple object identification [18]. Defined by the International Telecommunications Union a nd adopted by Internet SNMP and LDAP communities, an OID is persistent once assigned and its lifetime is infinite. OIDs are unique because they follow a distributed management model like the Internet Domain Name System, and are simple because they are a series of short integers. It's also possible to trace any level of an OID back to its owner.

An OODT information architectural principle suggests three phases in the development of metadata for the purpose of enabling heterogeneous data system interoperabi lity. The first phase is simply the identification of individual data system data dictionaries. Data dictionaries describe the elements and associated values used to describe the information or data model associated with a data system.

The second phase o f development involves the transforming of the data dictionary's native representation into a standard representation based on the ISO/IEC 11179 specification. This standard representation is used to describe each data element identified in the dictionary. The following XML fragment illustrates the use of XML to define a space science data element.

```
<?xml version="1.0"?>
<schema>
```

```
...
 <dataElement>
  <name>TARGET_NAME</name>
  <identifier>1.3.6.1.4.1.1306.2.10.997
  </identifer>
  <version>2001</version>
  <registrationAuthority>NASA.PDS
  </registrationAuthority>
  <context>Metadata.DataDictionary.Element
  </context>
  <definition>Identifies a target, which may be
   a planet, satellite, ring, region, feature,
   asteroid or comet.  See TARGET_TYPE.
  </definition>
  <dataType>CHARACTER</dataType>
  <format>N/A</format>
  <unit>none</unit>
  <maxSize>30</maxSize>
  <obligation>Mandatory</obligation>
  <maxOccurrence>1</maxOccurrence>
 </dataElement>
...
</schema>
```

The third phase of metadata development involves the manual analysis and comparison of data elements both within and across domains to identify or derive Common Data Elements (CDEs). This typically involves identifying similar element names or definitions, determining core concepts, possibly generalizing the concept, and determining key names and aliases. The resultant discipline, application, or enterprise specific CDEs enable higher levels of data system interoperability once adopted by individual data systems.

The OODT profile server described in Section 3 manages metadata descriptions of resources across and within distributed data systems. The metadata descriptions are created from and validated against the metadata registry and are packaged as "profiles." As mentioned before, resources can be almost anything including subsystems, services, data volumes, data collections, or data items. Examples of space science data items include images, spectra, and documents.

The OODT profile is an XML document that uses domain metadata to describe the data resources that exist within a data system. Profiles promote interoperability between data systems by providing a common structure and interchange language within which to describe system resources. The primary purpose of an OODT profile is to provide a resource description that is sufficient for a query manager to determine if the resource can resolve a query. This subsequently limits the number of resources that will have to consider the query. For example, within space sciences, a query for images of Jupiter should not have to be handled by the resource that maintains the Mars Global Surveyor images of Mars. A profile can be defined as a proper subset of the metadata that describes a resource and which is sufficient to determine whether the resource could resolve a query.

In addition to using XML as the interpeer exchange format, we also use XML as the common language for the OODT profiles. The advantages of XML include (1) superior expressiveness to HTML by allowing information-structure specifications, (2) simplicity compared to SGML in use and syntax, (3) wide acceptance as an Electronic Data Interchange (EDI) standard, and (4) the most compelling, its flexibility. The flexibility of XML allowed us to develop a generic structure for managing resource descriptions from any science domain.

The Document Type Definition (DTD) specification listed below illustrates the basic components of the resource profile. The DTD specification has three parts: the profile attributes, resource attributes, and the profile elements. We're using a DTD specification since the technology is widely supported. We are considering XML-Schema since it is now a recommendation of the World Wide Web Consortium (W3C).

```
<!ELEMENT profiles (profile*)>
<!ELEMENT profile (profAttributes, resAttributes,
 profElement*)>
<!ELEMENT profAttributes (profId, profVersion?,
 profType, profStatusId, profSecurityType?,
 profParentId?, profChildId*, profRegAuthority?,
 profRevisionNote*, profDataDictId?)>
<!ELEMENT resAttributes (Identifier, Title?,
 Format*, Description?, Creator*, Subject*,
 Publisher*, Contributor*, Date*, Type*, Source*,
 Language*, Relation*, Coverage*, Rights*,
 resContext+, resAggregation?, resClass,
 resLocation*)>
<!ELEMENT profElement (elemId?, elemName,
 elemDesc, elemType?, elemUnit?, elemEnumFlag*,
 (elemValue* | (elemMinValue, elemMaxValue)),
 elemSynonym*, elemObligation?,
 elemMaxOccurrence?, elemComment?)>
```

The profAttributes section of the XML document contains the attributes of the profile itself. The profId attribute provides a system-wide unique identifier for a profile instance and is currently being implemented as an OID. The profTitle and profDesc attributes provide descriptions of the profile where profTitle is more terse and appropriate for more frequent display in user interfaces. The profType attribute, defined further below, identifies the profile type. The profDataDictId attribute provides the identifier of the controlling domain data dictionary. The profChildId attributes allow for the creation of a profile hierarchy.

The resAttributes section of the XML document generically describes the resource using the 15 recommended elements of the Dublin Core initiative. A description of this initiative and the elements is available at http://purl.org/DC/. We've added three additional resource attributes. The resContext

element identifies the application environment or discipline within which the resource originates and is derived from a taxonomy of science disciplines. For example, NASA.PDS.Geoscience is used to indicate that the resource is associated with the Geoscience node of the Planetary Data System. The resLocation provides the location of the resource, typically represented as a URL. Finally, the resClass element identifies the resource within a taxonomy of resource types. Examples values are system.productServer, application.interface, data.granule and data.dataSet.

The profile element section of the XML document describes the data content that the resource manages using domain specific metadata. For example, the Planetary Data System (PDS) maintains an inventory of all science data sets that have been archived in the system. Within this inventory, the data sets are indexed on instrument and target body identifiers involved in the observation. These elements would be included in a profile's element section as illustrated below and indicate that this data set can resolve queries that have PDS instrument and target identifiers as query constraints.

As illustrated in the XML fragment below, each data element is defined using the meta-attributes elemId, elemName, and elemValue. These meta-attributes are consistent with those in the OODT data element registry to allow for reference and validation.

```
<profile>
 <profAttributes>
  <profId>1.2.3.4...</profId>
  <profType>PROFILE</profType>
  <profStatusId>ACTIVE</profStatusId>
 </profAttributes>
 <resAttributes>
  <Identifier>1.2.3.5...</Identifer>
  <Title>Viking Orbiter 1...</Title>
  <Format>text/html</Format>
  <Language>en</Language>
  <resContext>NASA.PDS</resContext>
  <resClass>data.dataset</resClass>
  <resLocation>http://pdscat...</resLocation>
 </resAttributes>
 <profElement>
  <elemId>1.2.3.6...</elemId>
  <elemName>INSTRUMENT_N...</elemName>
  <elemType>ENUMERATION</elemType>
  <elemValue>VISUAL IMAG...</elemValue>
 </profElement>
 <profElement>
  <elemId>1.2.3.7...</elemId>
  <elemName>TARGET_NAME</elemName>
  <elemType>ENUMERATION</elemType>
  <elemValue>MARS</elemValue>
  <elemValue>PHOBOS</elemValue>
  <elemSynonym>ADS.OBJECT_ID</elemSynonym>
 </profElement>
</profile>
```

The final phase of profile development involves implementing instances of profiles for specific domain resources. As is evident, the success of the distributed resource location concept is dependent on the existence of domain metadata captured in repositories such as data (element) dictionaries. Within such privileged domains, the registration of resources with the service is readily accomplished by extracting the necessary metadata from the domain's metadata repository, creating the resource profile, and then registering the profile with a profile server. This enables the successful location of resources within a domain.

Supporting interoperability between resources from different domains is strongly dependent on metadata compatibility, or how well the metadata spans the domains. For example, two related domains such as planetary science and astrophysics both associate one or more target bodies with most data products. However, unless the same identifiers are used for a specific target or a mapping between identifiers is determined, the attribute will not support resource location much less interoperability across the domains. In fact, as more sophisticated interoperability—such as data transformation and correlation—are requested, deeper levels of metadata compatibility will be required. For example, once a target body is identified, sufficient metadata must be available for coordinate system conversion.

The development of cross-discipline metadata is typically an arduous manual effort performed by teams of domain experts. However there are research efforts focusing on automatic methods for identifying common metadata across domains.

5. Application to the Planetary Data System

The Planetary Data System (PDS) manages and archives planetary science data for NASA's Office of Space Science. In existence since the late 1980s, the PDS developed—early on—a standards architecture that included a formal enterprise model, a means for collecting and associating metadata with science data products, and a peer review process for ensuring data and metadata validity. This active science data archive currently has over five terabytes of data stored and distributed on CD and DVD media. The standards architecture has proven to be the single most important element in maintaining consistency across the various science domains represented in the planetary science community and the six geographically distributed science discipline nodes of the PDS.

The primary focus of the PDS to-date has been the creation of a long-term science data archive on physical media. PDS provides community access to the data products primarily

through the appropriate distributed science discipline node. For example, the PDS imaging node is responsible for imaging data products and builds search and retrieval interfaces for specific data sets as requested by their users. Similarly, other discipline nodes are responsible for the data within their domain and build search and retrieval interfaces as needed. The guiding philosophy was that scientists working in the planetary science community would know what node or interface could provide the data of interest. This development approach resulted in heterogeneous search and retrieval systems and as the number of data sets increased, an exponential increase in the number of point-to-point interfaces.

The advent of the Web, a huge increase in potential users, and the large data volumes proposed for future solar system explorations missions has re-focused attention on distributed data management issues. These include a need for an integrated view of all the data in the archive, an integrated spatial/temporal access across the entire archive, and infrastructure support for science analysis and knowledge discovery.

In 1998, the PDS developed the prototype Distributed Inventory System (DIS) to inventory the resources across the PDS. This prototype took advantage of the wealth of metadata in the archive to described both data and non-data resources. The DIS described each resource in a label using pertinent metadata from the archive. The DIS uses the Object Description Language (ODL), a keyword/value language, to encode the metadata. URL/FTP links were included for all online resources. The labels were collected into a simple database. DIS engineers developed a simple Perl CGI-script search engine to search the label database. Results are provided as an HTML document for display in a browser.

This prototype was a success and provided the PDS with several lessons. First, it identified the PDS metadata as the key to the integration problem since it was sufficient to identify and describe both data and non-data resources, provided the necessary search attributes for finding resources, and was sufficient to determine whether a query could be resolved by the resource. In addition the prototype showed that the increase of point-to-point interfaces could be reduced to a linear function. The prototype also made certain limitations obvious. These were that ODL was not a universal interchange language, resource heterogeneity was still visible to the user, navigation and access was limited to HTTP links, and node data system interoperability is not supported.

In late 1999, PDS and OODT staff met to discuss the applicability of OODT's data management technology to PDS issues and PDS's information architectural concepts (data modeling and metadata standards) to OODT goals. One result of this collaboration is called the Next Generation DIS. The goals of this new system includes adopting a common interchange language (XML), hiding heterogeneity using encapsulation, using message passing protocol in a component based system architecture, maintaining system location independence, and providing a scaleable and extensible solution. Figure 7 illustrates the proposed system which will be prototyped using the Mars Global Surveyor (MGS) science data archives.

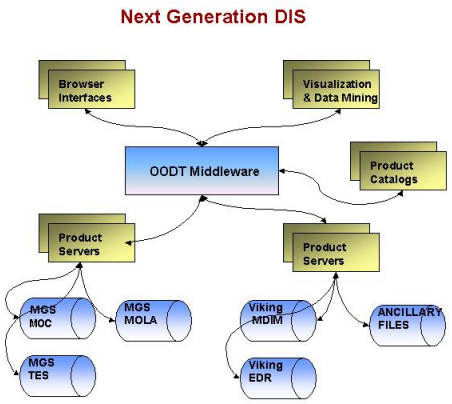

Figure 5 - Next Generation DIS—Mars Global Surveyor Prototype

OODT profiles will replace the ODL resource labels of the initial DIS system. A portion of an OODT resource profile for the Viking Mars Imaging Digital Image Model data set (Mars Mosaic created using original Viking images) is illustrated below. This profile provides sufficient information for a query handler to determine whether the data set could resolve an incoming query. In addition, this profile could also be easily modified to describe a product server that serves individual images by simply changing the resource description since the metadata encoded in the profile element section is applicable to both resources

```
<profile>
 <profAttributes>
  <profId>1.3.6.1.4.1.1306.2.102</profId>
  <profType>PROFILE</profType>
 </profAttributes>
 <resAttributes>
  <Identifier>VO1/VO2-MOSAIC-5-DIM-V1.0
  </Identifer>
  <Title>VO1/VO2_MARS_VISUAL</Title>
  <Format>text/html</Format>
  <resContext>NASA.PDS</resContext>
  <resClass>data.dataSet</resClass>
  <resLocation>http://pdsproto.jpl.nasa.
  </resLocation>
 </resAttributes>
 <profElement>
  <elemName>TARGET_NAME</elemName>
  <elemType>ENUMERATION</elemType>
  <elemValue>MARS</elemValue>
 </profElement>
 <profElement>
  <elemName>MAXIMUM_LATITUDE</elemName>
  <elemType>REAL</elemType
```

```
  <elemMinValue>87.50000</elemMinValue>
  <elemMaxValue>90.00000</elemMaxValue>
 </profElement>
</profile>
```

The initial goal of the prototype will be to support the creation of a data coverage plot as illustrated in Figure 6. In this plot, indicators showing the geographical location of MGS imaging (red), altimeter (blue), and thermal emission spectrometer (green) data products are overlaid onto a Mars image mosaic from the Viking Mars image data set. Four data sets at several geographically distributed data systems will be accessed to produce the plot.

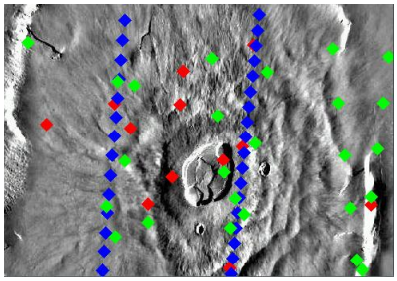

Figure 6 - Mars Global Surveyor Coverage Plot

6. Application to the Deep Space Network

Established in 1958, NASA's Deep Space Network (DSN) is a global network of antennas that communicate with spacecraft. The DSN facilitates interplanetary spacecraft missions and several Earth-orbiting missions. It is currently made up of three facilities across the world: at Goldstone, in California's Mojave Desert; near Madrid, Spain; and near Canberra, Australia[3].

The DSN plays an integral role in the success of missions. Amongst a few of its duties are: acquiring telemetry data from spacecraft, transmitting commands to spacecraft, gathering science data, and tracking spacecraft position and velocity[3]. However, these tasks are hindered by the distributed and heterogeneous nature of the DSN. These obstacles complicate data sharing and mining, data modeling, and the creation of a common information architecture.

The OODT framework directly addresses the key issues of the DSN. OODT's emphasis on implementation of both a component framework along with the identification of local data dictionaries and models for each system provides an excellent vehicle for establishing an information architecture for the DSN. In addition, OODT's design principle of hiding local system interfaces through data abstraction allows for the implementation of a common information architecture. OODT allows for data elements tied to local information systems to be translated into common data elements described for the entire DSN. Its extensible architecture provides the DSN with what could be considered "plug and play" data management. In addition, this architecture provides a means by which data and local agents can be located, retrieved or executed on any number of systems and machines using a common XML exchange structure as defined in section 3. This distributed computing model fits very nicely into the overall DSN architecture.

As proof of concept, we created and launched a prototype for the DSN. The first phase of this prototype involved creating the "Mission and Ground Asset Database." This database provides a basic description of missions, ground stations, spacecraft, and ground antennae. It was created to provide an online reference of key mission parameters and values and to provide a long term vision for tying together capabilities of ground stations around the world. Figure 7 shows a high level entity/relationship diagram of the database. This model will allow engineers to discover possible constraints of the current ground systems and or missions. In addition, the database could be used to understand how to optimize workflows in and around ground stations.

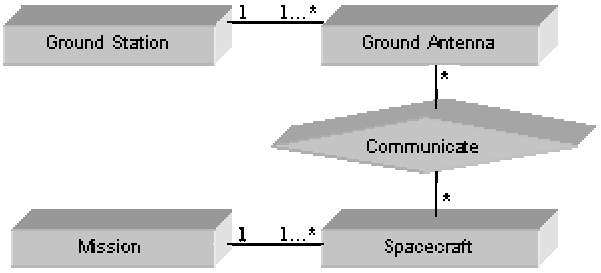

Figure 7 - ER Diagram of Mission and Ground Asset Database

The implementation of the database in conjunction with the OODT architecture allowed for the ground station information captured to be published and exchanged over the Internet using an XML interface. Not only is the database now searchable with in the architecture, it can be discovered by other systems by passing the XML structure to the OODT system. OODT used a product server to query and return the results of the database using the OODT XML query definition. The product server was defined in a profile which describes the missions and ground stations that are maintained by the database. This allows for a query of mission information to first locate a database which contains the desired information. Looking forward, this will allow for multiple databases to be tied together from disparate ground asset databases around the world if a standard XML structure can be defined for querying that data. An excerpt of the XML structure from the profile can be seen below.

[3] http://deepspace.jpl.nasa.gov/dsn/

```
<profile>
 <profAttributes>
  <profId>1.3.6.1.4.1.1306.5</profId>
   <profVersion>1.0</profVerision
  <profType>PROFILE</profType>
 </profAttributes>
 <resAttributes>
 <Identifier>JPL.TMOD.PRODUCT_SERVER</Identifier>
  <Title>Mission and Ground Asset DB</Title>
  <Format>text/html</Format>
  <resContext>NASA.JPL.TMOD</resContext>
  <resClass>application.interface</resClass>
  <resLocation>http://velcro.jpl.nasa.gov...
  </resLocation>
 </resAttributes>
 <profElement>
  <elemName>MISSION_NAME</elemName>
  <elemType>ENUMERATION</elemType>
  <elemValue>Cassini</elemValue>
  <elemValue>Galileo</elemValue>
 </profElement>
 <profElement>
  <elemName>GROUND_STATION_LOCATION</elemName>
  <elemType>ENUMERATION</elemType>
  <elemValue>Goldstone</elemValue>
  <elemValue>Canberra</elemValue>
 </profElement>
</profile>
```

JPL's Interplanetary Network and Information Systems Standards Program has initiated several efforts to explore the use of XML as a means to establish common data definition and access mechanisms for a reference mission information architecture. This architecture is being worked with support from the Consultative Committee on Space Data Systems through an XML Working Group, and the Object Management Group's Space Domain Task Force. The goal is to establish world-wide mechanisms for exchanging mission information as well as supporting the use of XML throughout the mission life cycle. OODT is continuing to work on supporting this vision and is now working to allow other databases to be queried using OODT's XML components. XML and OODT play a critical role in helping both the DSN and other efforts from NASA and other agencies establish architectures for the capture, location, and exchange of critical mission information across the boundaries of distributed and heterogeneous systems.

7. FUTURE

We've realized the physical deployment of the current OODT framework using CORBA as the network substrate. CORBA's object model represents servers as interfaces with methods, while clients call those methods. The logical deployment, however, is different. It is a set of interconnected peers that pass around a query object. Each peer examines the query and may add any results that are relevant. The query service, described in Section 3, simulates this P2P architecture by crawling the network of servers on behalf of the user and maintaining the query object.

By replacing the CORBA substrate with a P2P substrate, we can better leverage P2P features. For example, the query service completely disappears because its role is redundant in a P2P architecture. In P2P, there are no clients or servers, just peers. Peers publish service advertisements that describe the functions they can carry out. Profile peers would publish advertisements for resource location services. Product peers would publish advertisements for data retrieval services. In this model, a user would "inject" a query into a system. Service advertisements would indicate what peers would be capable of handling the query. Results would come back to the user's peer asynchronously; and the user can terminate the query as soon as s/he receives the desired results.

Furthermore, a P2P substrate addresses scalability. In the CORBA model, the query service must wait its turn when running a query on behalf of a user at an overburdened profile or product server. In the peer model, a popular data center can be leveraged by providing more than one peer at the center to handle requests. These peers may be located on multiple CPUs and network interfaces to maximize concurrency and availability. In essence, a system administrator can speed up a site by starting more peers.

In addition, we are focusing on developing new components in the architecture to manage metadata by creating a metadata service. The purpose of the metadata service is to provide a series of registries that allow for the management of data dictionaries, data elements and resource descriptions. This service will be designed around three major capabilities: capturing data dictionary schemas, capturing data elements and capturing resource definitions for data systems, data sets, and data products. We see this as critical to the validation of profiles for describing system resources.

OODT is also continuing to work across disciplines. We are working with the National Institutes of Health to explore the use of OODT for capturing and sharing research produced in disparate research laboratories and are working with JPL Institutional Computing and Information Services program (ICIS) to explore infusion of data management services and information architectures for describing, building and operating distributed data systems at both the institutional and project level.

The success of OODT has in many ways been driven by the desire to share information. While there are many technical, social and legal issues related to data sharing and interoperability, the prospect of deriving new knowledge through data correlation is immensely high. In addition, providing standard data management frameworks for building information management systems that can dynamically plug into larger information management systems is certainly a grand vision. One thing is clear: science and engineering plan to continue to produce large

volumes of data that are orders of magnitude larger than those already captured. The need to better capture, organize, describe, reuse and share information within and across disciplines will continue to be key information management needs and research topics in the foreseeable future.

8. REFERENCES

[1] Tim Bray et al. Extensible Markup Language (XML) 1.0 (Second Edition), Cambridge: World Wide Web Consortium, 2000.

[2] Emelie Rutherford. "The P2P Report," CIO Magazine, January 2000.

[3] "White Paper: A National Virtual Observatory for Data Exploration and Discovery," October 1999.

[4] Li Gong. Project JXTA: A Technology Overview, Palo Alto: Sun Microsystems, 2001.

[5] Erich Gamma et al. *Design Patterns: Elements of Reusable Object-Oriented Software*, Reading: Addison-Wesley, 1995.

[6] Object Management Group. *The Common Object Request Broker: Architecture and Specification*, Needham: OMG, Inc., 2001.

[7] Grady Booch et al. *The Unified Modeling Language User Guide*, Reading: Addison-Wesley, 1999.

[8] Ken Arnold, James Gosling, and David Holmes. *The Java Programming Language*, Reading: Addison-Wesley, 2000.

[9] J. Steven Hughes et al. "A Multi-Discipline Metadata Registry for Science Interoperability," Santa Fe: Open Forum on Metadata Registries, ISO/IEC JTC1/SC32, Data Management and Interchange, 2000.

[10] Tim Berners-Lee et al. "Uniform Resource Identifiers (URI): Generic Syntax (RFC2396)," Reston: The Internet Society, 1998.

[11] Xerox, Inc. DocuShare: Knowledge Sharing Software, Rochester: Xerox, Inc., http://docushare.xerox.com/

[12] Y. Goland et al. "HTTP Extensions for Distributed Authoring—WEBDAV (RFC2518)," Reston: The Internet Society, 1999.

[13] Ned Freed and Nathaniel Borenstein. "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies (RFC2045)," Reston: The Internet Society, 1996.

[14] Roy Fielding et al. "Hypertext Transfer Protocol—HTTP/1.1 (RFC2616)," Reston: The Internet Society, 1999.

[15] Sun Microsystems. "Java Servlet Specification 2.3," Palo Alto: SMI, 2001.

[16] ISO/IEC 11179 - Specification and Standardization of Data Elements, Parts 1-6, ISO/IEC specification, http://www.iso.ch/iso.

[17] Dublin Core Metadata Initiative. The Dublin Core Element Set Version 1.1, Dublin: DMCI, July 1999.

[18] ITU-T. Information Technology—Abstract Syntax Notation One (ASN.1): Specification of basic notation (X.680), Geneva: International Telecommunications Union, 1997.

9. ABOUT THE AUTHORS

Daniel Crichton is a Project Element Manager at JPL and the principal investigator for the Object Oriented Data Technology task where he is leading a research effort developing distributed frameworks for integrating science data management and archiving systems. He also currently serves as the implementation manager and architect of a JPL initiative to build an enterprise data architecture. His interests are distributed architectures, enterprise and Internet technologies, and database systems. He holds a B.S. and M.S. in Computer Science. He can be reached at Daniel.Crichton@jpl.nasa.gov.

Steven Hughes is a System Engineer at JPL and a Co-Investigator for the Object Oriented Data Technology task. He is currently the technical lead engineer for the Planetary Data System and was instrumental in the development of the planetary science metamodel. His interests are in distributed architectures and the role of metadata in interoperability. He holds a B.S. and M.S. in Computer Science. He can be reached at Steven.Hughes@jpl.nasa.gov.

Sean Kelly is an Independent Consultant based in Silicon Valley who provides expertise in object oriented software engineering, distributed systems, Internet technologies, and structured information. In addition to consulting for JPL, he writes a regular column on XML technologies for *Inside XML Solutions*. He holds a B.S. in Computer Science and a B.S. in Technical Communication. He can be reached at Sean.Kelly@jpl.nasa.gov.

Paul Ramirez is a Software Associate at JPL. He is currently supporting integration of the Object Oriented Data Technology task. His interests are in software architectures, database systems, and web technologies. He holds a B.S. in Computer Science from California Polytechnic University, Pomona. He can be reached at Paul.Ramirez@jpl.nasa.gov.